# Relevance of An Accuracy Control Module - Implementation into An Economic Modelling Software

Buda, Rodolphe

GAMA - Université de Paris 10 Nanterre

2005

# Relevance of An Accuracy Control Module Implementation into An Economic Modelling Software

Rodolphe BUDA
*GAMA, University of Paris 10*

2005

---

**Summary**

*Economic modelers or econometricians know that they have to control the accuracy of the computations, to get their results safe. Unfortunately, even if a few softwares allow to control it (especially in using an interval arithmetic), most of them are presented like black boxes, from an arithmetical point of view. We have tried, in this paper, to provide the point of view of a macroeconomic modelling software's builder. After having recalled the accuracy problem and its solutions (algebraic and arithmetical), we'll present the* `GNOMBR` *software we have developed, based on multiple precision arithmetic methods. Then we'll present an application to an input-output model. Then we'll give elements to conclude to the relevance of this technique.*

**Key-words** : Modelling software - Floating point arithmetic - Accuracy results control - Arithmetic accuracy computer

**J.E.L. Classification** : C82, C87, C88.

---

## 0. - Introduction

This paper collects the reflections we have had during the building of a multi-dimensional[1] economic modelling software (`SIMUL` *i.e.* **S**ystème **I**ntégré de **M**odélisation m**UL**tidimensionnelle)[2]. This software usually can estimate multi-dimensional econometric equations quickly and solve the whole system[3]. So it's divided in some modules (estimation, data bank managment and so on). During this work, we have had a reflection about the arithmetic accuracy trouble. Hence, we have decided to build a specific arithmetic tool (`GNOMBR` *i.e.* **G**rands **NOMBR**es). The implementation of another arithmetic into an economic model is not new[4]. However, we provide here the double economist and software developer point of view. Furthermore, our tool is a multiple precision arithmetic implemented in Turbo-Pascal 7.0 available in two versions : library and direct Dos prompt executable version.

We have divided this paper into two parts. In a first part, we'll recall the arithmetic accuracy problem. Then we'll present the classical solutions and the recent arithmetical substitution solutions. We'll focus our analysis on the `GNOMBR` software. We'll especially present the implementation of it in an input-output simplified model. Furthermore, we'll present some results and some algorithms related to this software building. Finally, we'll try to conclude about the usefulness of such tools, and about the economic's domain where the arithmetic accuracy tools seem to be the more relevant.

## 1. - The arithmetic computer's accuracy problem

In this first part, we'll explain why the standard arithmetic computer - the floating point arithmetic or `IEEE-754` Standard - and even its substitutes, will never completely satisfy the accuracy of computation. Indeed, we have to maximize the level of accuracy, in ordering elementary operations in the formulae (algebraic method) or to substitute the standard arithmetic computer by other arithmetics (arithmetic method).

### a - The lack of representation of the floating-point arithmetic

One tells, when J. von Neumann considered the first results of the ENIAC computer, he was very disappointed because of its lack of accuracy. This problem comes from the unperfect arithmetic representation of the digits. All computers have got this trouble[5] : ordinary personal computers, super

1. - Multi-regional (r), multi-sectoral (s), multi-period (t) analysis.
2. - The building of this software is integrated in the economic thesis we are finishing, "Modélisation multi-dimensionnelle et analyse multirégionale de l'économie française" under the supervision of the Professeur Raymond Courbis.
3. - We only need an instruction to implement such equations : $Y_t^{r,s} = X_t^{r,s}.a^{r,s} + \varepsilon_t^{r,s}$.
4. - Such way has been already used (M.E.Jerrell, 1997) with the interval arithmetic.
5. - We present the error of computation of factorial function in Appendix.

computers (see eg.1) and formal computation is affected too (see eg.2).

**#1** - The equation system [6]

$$\begin{cases} r = 9x^4 - y^4 + 2y^3 \\ x = 10864 \\ y = 18817 \end{cases}$$

provides $r = 0$ with a **CDC-7600** computer, $r = 2$ with a **CRAY ONE** computer, and $r = 7.081589E+08$ with a **VAX** computer, but the exact result is $r = 1$.

**#2** - The series [7]

$$\begin{cases} u_0 = 2 \\ u_1 = -4 \\ u_{n+1} = 111 - \frac{1130}{u_n} + \frac{3000}{u_n.u_{n-1}} \end{cases}$$

provides

$$u_{90} = 99.998040316541816165\ldots$$

with Maple software, but the exact result is

$$u_{90} = 6.0000000996842706405\ldots$$

**Fig.1 - Two examples of accuracy troubles during computation**

The trouble comes from the unperfect computer's arithmetic, the floating-point arithmetic. It's impossible to represent the whole properties of the real numbers with this arithmetic, especially continuity property. According to a mathematical point of view, it's a nonsense to actually compute a limit or a derivation function. $\forall$ element $X \in \mathbb{R}$ represented by the floating element [8] $\tilde{X}$ which belongs to $\mathbb{F}$, we have

$$\tilde{X} = s.M.b^e$$

where $s$ is the sign of $\tilde{X}$, $M$ the mantissa encoded with $t$ digits which belong to the $b$ base, and $e$ is the exponent. Because of the finite size of the mantissa, during calculation, the computer truncates the numbers [9].

6. - From M.Pichat & J.Vignes (1993, pp.3-5).
7. - From M.Daumas & J.M.Muller (Eds.) (1997, pp.2-8).
8. - See M.La Porte & J.Vignes (1974, *Algorithmiques numériques - Tome 1*, Paris, Technip, 226 p. For an axiomatic presentation, see D.E.Knuth (1997, pp.229-45), especially pp.17-39.
9. - See J.Berstel *et al.* (1991, *op.cit.*, pp.123-204). See Pascal programs of N.Wirth (*op.cit.*, pp.94-100) who suggested to use binary digits.

Thus, we encounter two kinds of problems :

1°- the computer can't reach the infinite limits of the real numbers ensemble (the greatest positive number is 1.0E+40, but it's not infinite [10].

2°- the computer can't represent the separability property of numbers : `1.0E+40 + 1 = 1.0E+40`. Unfortunately, a computer doesn't follow a systematical rule of rounded numbers. We cant expect if truncation of the real number will overestimate or underestimate it. Both numbers `E.FFFFFF47 and E.FFFFFF54` will be truncated in `E.FFFFFF5`. However, the number of significantly digits follows a Laplace-Gauss statistic rule [11]

Furthermore, the trouble is not the same for all operators [12] and algorithms. M.Pichat and J.Vignes (*op.cit.*) have classified three cases :

1° - "Finish algorithms" : The solution computed should be exact. From matrix algebra computation (Gauss-elimination methods...).

2° - "Iterative algorithms" : The solution belongs to a convergence interval. From matrix algebra computation (Gauss-Seidel, Newton methods...). The algorithm converges if $F(X_q) = \underline{0}$ ; stationary if $|X_q - X_{q-1}| = 0$ ; divergent if $q > N_{n_{max}}$ where $F$ represents the algorithm applied to the variable $X$ during $q$ iterations, and $\underline{0}$ represents the data process zero, significantly near from his mathematical value.

3°- "Approached algorithms" : The solution depends on an interval of discretization [13]. From the Differential calculus.

### b - The main algebraic control method

We here expose the two main algebraic control methods, which are the *Conditioning method* and the *Horner's rule*.

---

10. - We have to specify here, that until the sixties, the arithmetical representation of infinitesimal numbers were sometimes paradoxical. To correct this problem, A.Robinson (*Non-Standard Analysis*, North-Holland, Amsterdam, 1966), has divided $\mathbb{R}^+$ into three parts of numbers : the *ideally little numbers* (i-little), the *appreciable numbers*, and the *ideally great numbers* (i-great) - see M.Diener & A. Deledicq, *Leçon de calcul infinitésimal*, Paris, A.Colin, 1989. Such innovation should provide another way of progress toward better accuracy of arithmetic computers.

11. - See M.Maillé (1982).

12. - Division operator deteriorates more the accuracy of the results than multiplication, addition or substraction.

13. - See J.Dumontet & J.Vignes (1989).

*(i) The conditioning method* : Conditioning a system $X.a = b$ means we compute the following formula $Cond(X) = \frac{\sqrt{\lambda_{max}}}{\sqrt{\lambda_{min}}}$, where $\lambda_i$ are the eigenvalues of the matrix $X$, to know if the disturbance of the system during his transformation from $X.a = b$ to $(X + \Delta.X).a = (b + \Delta.b)$ is significant or not. The matrix is well conditioned if $Cond(X) \sim 1$ or we have to change the pivot.

*(ii) The Horner's rule* : This method [14] consists to change such polynomial formula $P(x) = \sum_{i=1}^{n} a_i x^i$ according to this way :

$$P(x) = a_n.x^n + a_{n-1}.x^{n-1} + \ldots\ + a_1.x^1 + a_0.x^0$$

$$\Rightarrow\ P(x) = x(x(\ldots x(a_n x + a_{n-1}) + \ldots)) + a_1) + a_0$$

```
Function EVAL( P :Polynome ; X :Real ; N :Integer) : Real ;
Var i :Integer ;
  r :Real ;
Begin
  r⟵P[n] ;
  For i⟵n-1 DownTo 0 do EVAL⟵r*P[i] ;
End ;
```

**Fig.2 - Horner's rule Algorithm**

### c - The arithmetic substitution methods

We can divide the search about arithmetic substitution into different kinds. The first kind keeps the basis 10 and the basic arithmetical operators and the mantissa. Other methods [15] provide a greater accuracy arithmetic. The second kind of methods changes the arithmetic and/or the basic operators.

14. - W.G.Horner presented this method in a paper in connection with a procedure for calculating the polynomial roots (*Philosophical Transactions*, Royal Society of London, 109, 1819, pp.308-335).
15. - See U.Kulisch & W.N.Miranker (1981)

*(i) The stochastic arithmetics* : `CESTAC`[16] method (**C**ontrôle et **E**stimation **ST**ochatique des **A**rrondis de **C**alculs) tries to estimate the number of significantly digits of a number by studying the round's error diffusion, in evaluating the parameters of the error's probability distribution. So, the unobservable digits of the rounded number are simulated[17].

*(ii) The interval arithmetics* : This method begin to check the validity interval of the computation[18], then it makes calculation on interval $[a] \rightarrow [\underline{a}, \overline{a}]$ (not on numbers $a$) according the following operator's rules :

**Table 1 : Interval algorithm operator's rule**

| | | |
|---|---|---|
| `Addition` | $[a] + [b]$ | $[\underline{a} + \underline{b}, \overline{a} + \overline{b}]$ |
| `Soustraction` | $[a] - [b]$ | $[\underline{a} - \underline{b}, \overline{a} - \overline{b}]$ |
| `Multiplication` | $[a][b]$ | $[Min(\underline{ab}, \underline{a}\overline{b}, \overline{a}\underline{b}, \overline{a}\overline{b}), Max(\underline{ab}, \underline{a}\overline{b}, \overline{a}\underline{b}, \overline{a}\overline{b})]$ |
| `Division` | $[a]/[b]$ | $[\underline{a}, \overline{a}][\frac{1}{\overline{b}}, \frac{1}{\underline{b}}]$ `si` $0 \notin [b]$ |

The lazy rational arithmetic is an alternative of the interval one. It tries to avoid the using of rational number during computation[19].

*(iii) The multiple precision arithmetic* : Such method involves that the number of significant digits if very high[20]. Consequently, the round of the numbers happen behind a lot of digits. The accuracy of the number increases. However, such arithmetic needs to implement new arithmetic operations ($+ - / *$ and ˆ). Multiple precision arithmetic is available in a lot of libraries translated in languages[21] and macro-languages[22]

16. - We could translate it with "Stochastic computation's rounds estimation and control" - see M.Pichat & J.Vignes (1993, pp.170-175).

17. - See M.Pichat & J.Vignes, *op.cit.*, pp.89-189 ; and M.Daumas & J.M.Muller (Eds.), *op.cit.*, pp.65-92)

18. - See G.Alefeld (1983). For an implementation of interval arithmetic to an input-output model, see M.E.Jerrell (1997).

19. - See M.Daumas & J.M.Muller (*op.cit.*, pp.126-29).

20. - See R.E.Moore (1966). D.E.Knuth (*op.cit.*, pp.265-94) and U.Kulisch (2000.a, 2000.b) (resp.) have translated this arithmetic in Assembler and Fortran (resp.). See J.Berstel et al. (*op.cit.*, 1991, tome 2, pp.149-204) and M.Daumas & J.M.Muller (Eds.) (*op.cit.*, pp.93-116) too.

21. - Let us quote GrandsNombres in Turbo-Pascal (J.Berstel et al., *op.cit.* 1991) ; MP in FORTRAN, MP GNU in C, MP FUN in C, PARI in C/C++ et RANGE en C++.

22. - M.Sofronioua & G.Spaletta (2005) have implemented their arithmetic with Mathematica.

*(iv) The exact arithmetic* : This method computes without error neither round step. However only integer or rational numbers can be used during the procedure. This method is unfortunately slower than the other methods[23].

*(v) The algebraic arithmetic* : This method uses Bezout's Theorem. (if $A$ and $B$ are prime numbers each, ($\exists\, U, V \ / \ U.A + V.B = 1$). If we consider the polynomial relationship

$$A(X) = Q(X).\phi(X) + R(X)$$

and $a = A(\alpha)$ and $b = B(\alpha)$ hence we can define arithmetical operations[24] :

**Table 2 : Algebraic arithmetic operations**

| | | |
|---|---|---|
| `Addition` | $a+b$ | $A(\alpha) + B(\alpha) = (A+B)(\alpha)$ |
| `Substraction` | $a-b$ | $A(\alpha) - B(\alpha) = (A-B)(\alpha)$ |
| `Multiplication` | $ab$ | $A(\alpha).B(\alpha) = (AB \ \ mod \ \ \phi)(\alpha)$ |
| `Inverse` | $1/a$ | $U(\alpha)$ |

*(vi) Dynamical arithmetic* : We consider a numerical base $b$ where we don't use the digits $\{1, \ldots \ b-1\}$, but the digits $\{-a, \ldots \ +a\}$ where $a \ / \ a < b-1$. For example, the number $(1745)_{10}$ - read `1745` in base `10` - will be written [2][−3][4][5] or [2][−3][4][−5]. This method (A.Avizienis, 1961) avoids to manage some carry over.

23. - For a more complete presentation, see M.Daumas & J.M.Muller, op.cit., pp.117-36.
24. - See M.Daumas & J.M.Muller op.cit., pp.129-130.

## 2. - Presentation of a multiple precision arithmetic tool : `GNOMBR`

In this second part, we firstly provide an overview of the tool, then we'll show how we use it to program a simplified input-output model. Finally, we'll present the basic algorithms of the tools and some significant results we can obtain with it.

### a - A general presentation

We can increase the accuracy of a Turbo-Pascal program with the library of `GNOMBR`. `GNOMBR` uses the usual operators "+", "-", "*", "/" and "^", but unfortunately, it is only able to compute two operands at the same time. To compute large formulae, we have to use temporary variables and to divide it into a few formulae.

*In the example on the right side, we can observe that we have computed the exponentiation by 2 of a number which has a great accuracy concerning its decimal digits.*

*Directly from the DOS-prompt, the* `GNOMBR` *software provides the result according to his own arithmetic and according to the floating-point arithmetic, when both computations are possible.*

*Each number is displayed divided into five digits packages. Each package has got two parts : the mantissa digits and the exponent of the basis 10.*

*On the top of each number, according to the multiple precision arithmetics, we can find the total number of packages of the representation of the number.*

```
 CALCUL EN GRANDS NOMBRES

    10
   12334.0E+0005
   96554.0E+0000
     694.0E-0005
   45647.0E-0010
   54756.0E-0015
   34653.0E-0020
     324.0E-0025
   24324.0E-0025
     243.0E-0035
   20000.0E-0040

^ 2

----------------
    19
    1521.0E+0015
   51374.0E+0010
   87470.0E+0005
    7109.0E+0000
   38809.0E-0005
   18440.0E-0010
   88482.0E-0015
   70989.0E-0020
   92706.0E-0025
   77532.0E-0030
   18742.0E-0035
   72199.0E-0040
    4858.0E-0045
   17764.0E-0050
   63089.0E-0055
   34687.0E-0060
   91194.0E-0065
   19146.0E-0070
   24000.0E-0075

CALCUL EN VIRGULE FLOTTANTE

 1.23349655400694E+0009
^ 2
------------
 1.52151374874701E+0018
```

**Fig.3 - An example of `GNOMBR`'s use from MS-DOS prompt, with comment**

The size of the accuracy depends on the memory used for the variable of the program. The accuracy indeed decreases as soon as the number and/or the size of variables increases. For only one operation, the maximal accuracy is 5000 digits. Each number is represented by some packages of five digits. For example the number `1205040` becomes `12.E+0005 + 5040.E+0000`. The complete software is composed of a Dos-prompt computer (`CALC_GN`) and an editor (`EDIT_GN`). Each program gets parameters in configuration file. The editor translates the floating-point source code in a multiple precision source code. The `TABL_GN` program writes tables of results.

*(i) The Dos-prompt mode computer use* : The syntax is the following

```
CALC_GN # [Par_1][Arg_1] [Operator] [Par_2][Arg_2] [Par_3][N_file] [Com]
```

where `#` means we work directly from Dos-prompt, the `Par_i` are the great numbers files used. `Par_1` and `Par_2` use two located codes : *space bar code* or "`$`" and, `Par_3` uses two located codes : *space bar code* or "`@`" to assign an output file, the `Arg_i` are digital arguments (or file names) ; `Com` represents a comment. Finally, the `Operator` is a character `+ - * /` or `^`.

```
C :\>CALC_GN # 1.26636 * 65.23526
C :\>CALC_GN # $X.DAT + 65.23526
C :\>CALC_GN # $X1.DAT - $X2.DAT
C :\>CALC_GN # $X1.DAT - $X2.DAT @X1_X2.DAT
```

**Fig.4 - Some Dos-prompt uses examples**

We have presented a multiplication between 1.26636 et 65.23526, then an addition between a number inside the `X.DAT` file and the number 65.23526. Then a substraction between the number inside the file `X1.DAT` and the number inside `X2.DAT`. The result will be written in the `X1_X2.DAT` file.

We obtain the same results if we create an instructions file according to the same syntax[25] :

```
~LISTE DES OPÉRATIONS
~---------------------
~1.26636 * 65.23526
$X.DAT + 65.23526
$X1.DAT - $X2.DAT @X1_PL_X2.DAT CECI EST UN COMMENTAIRE
```

**Fig.5 - Some instructions executed from a file**

25. - The lines beginning by ~are considered as a comment.

*(ii) The program's implementation mode use* : This method is developed according to two steps [26].

1° - Firstly we have to create the Turbo-Pascal program (4.0 and the following). In this program, we have to translate each complex formula into some formulae. We are only allowed to use two variables and one operator per formula. If we have, for example, the initially formulae `VAR1 :=VAR2/(VAR3+VAR4) ;` we use temporary variable (`TEMP`) according to this following way :

```
TEMP :=VAR3+VAR4 ;
VAR1 :=VAR2/TEMP ;
```

2° - To specify the nature of the operation [27], we have to use some codes from the first column in the same line of the formula to be specified :
- `{%L}` specifies a reading operation,
- `{%B}` specifies a comparison operation (with a boolean result),
- `{%V}` specifies a value assignation,
- `{%A}` specifies the transfer of value from a variable to another,
- `{%C}` specifies a formula computation.

### b - An application to a simplified input-output model

In this section, we expose the implementation of multiple precision arithmetic `GNOMBR` in a simplified input-output model [28] programmed in Turbo-Pascal. The program of this model has been implemented so that it provided both kinds of results, floating-point and a multiple precision one.

*(i) The basic input-output model* : We have divided a fictive economy into three sectors. Let us consider $C$ the matrix of consumptions necessary to the production and $P$ the vector of production. Let's assume we want to simulate an increase of $p_2$ (production of the second sector) of 30 billion dollars, under a convergence threshold $s = 0.00009$.

26. - See R.Buda (1999).
27. - The specification are between accolades. We have used such codes because Turbo-Pascal consider them as comments. So these codes are only decoded by `GNOMBR`.
28. - According to the famous work of W.W.Leontief (*The Structure of American Economy*, Oxford University Press, 1940).

$$\mathtt{P}_{\mathtt{f}}^{\mathtt{i}} = \begin{pmatrix} 300.0 & 1000.0 & 600.0 \end{pmatrix}$$

$$\mathtt{C}_{\mathtt{f}}^{\mathtt{i}} = \begin{pmatrix} 150.0 & 10.0 & 30 \\ 35.0 & 390.0 & 80 \\ 15.0 & 100.0 & 90.0 \end{pmatrix}$$

where $P_f^i$ and $C_f^i$ (resp.) are the *ex ante* production vector and consumption matrix (resp.) in floating-point arithmetics[29].

$$\mathtt{P}_{\mathtt{f}}^{\mathtt{j}} = \begin{pmatrix} 3.00717000\mathrm{E}+0002 & 1.04669800\mathrm{E}+0003 & 6.04635000\mathrm{E}+0002 \end{pmatrix}$$

$$\mathtt{P}_{\mathtt{m}}^{\mathtt{j}} = \left(\begin{array}{rrr} 300.0\mathrm{E}+00000 & 1046.0\mathrm{E}+00000 & 604.0\mathrm{E}+00000 \\ 71700.0\mathrm{E}-00005 & 69799.0\mathrm{E}-00005 & 63500.0\mathrm{E}-00005 \\ & 99999.0\mathrm{E}-00010 & \\ & 99999.0\mathrm{E}-00015 & \\ & 99999.0\mathrm{E}-00020 & \\ & 99999.0\mathrm{E}-00025 & \\ & 99998.0\mathrm{E}-00030 & \\ & 80000.0\mathrm{E}-00035 & \end{array}\right)$$

where $P_f^j$ and $P_m^j$ (resp.) are the *ex post* production vector in floating-point arithmetics and multiple precision arithmetics (resp.).

$$\mathtt{C}_{\mathtt{v}}^{\mathtt{j}} = \begin{pmatrix} 1.50358500\mathrm{E}+0002 & 1.04669800\mathrm{E}+0001 & 3.02317500\mathrm{E}+0001 \\ 3.50836500\mathrm{E}+0001 & 4.08212220\mathrm{E}+0002 & 8.06180000\mathrm{E}+0001 \\ 1.50358500\mathrm{E}+0001 & 1.04669800\mathrm{E}+0002 & 9.06952500\mathrm{E}+0001 \end{pmatrix}$$

$$\mathtt{C}_{\mathtt{m}}^{\mathtt{j}} = \left(\begin{array}{r|r|r} 150.0\mathrm{E}+00000 & 10.0\mathrm{E}+00000 & 30.0\mathrm{E}+00000 \\ 35850.0\mathrm{E}-00005 & 46697.0\mathrm{E}-00005 & 23175.0\mathrm{E}-00005 \\ & 99999.0\mathrm{E}-00010 & \\ & 99999.0\mathrm{E}-00015 & \\ & 99999.0\mathrm{E}-00020 & \\ & 99999.0\mathrm{E}-00025 & \\ & 99999.0\mathrm{E}-00030 & \\ & 98800.0\mathrm{E}-00035 & \\ \hline 35.0\mathrm{E}+00000 & 408.0\mathrm{E}+00000 & 80.0\mathrm{E}+00000 \\ 8364.0\mathrm{E}-00005 & 21221.0\mathrm{E}-00005 & 61799.0\mathrm{E}-00005 \\ 99999.0\mathrm{E}-00010 & 99999.0\mathrm{E}-00010 & 99999.0\mathrm{E}-00010 \\ 99999.0\mathrm{E}-00015 & 99999.0\mathrm{E}-00015 & 99999.0\mathrm{E}-00015 \\ 99999.0\mathrm{E}-00020 & 99999.0\mathrm{E}-00020 & 99999.0\mathrm{E}-00020 \\ 99999.0\mathrm{E}-00025 & 99999.0\mathrm{E}-00025 & 99999.0\mathrm{E}-00025 \\ 99999.0\mathrm{E}-00030 & 99999.0\mathrm{E}-00030 & 99998.0\mathrm{E}-00030 \\ 52200.0\mathrm{E}-00035 & 53200.0\mathrm{E}-00035 & 45500.0\mathrm{E}-00035 \\ \hline 15.0\mathrm{E}+00000 & 104.0\mathrm{E}+00000 & 90.0\mathrm{E}+00000 \\ 3585.0\mathrm{E}-00005 & 66979.0\mathrm{E}-00005 & 69525.0\mathrm{E}-00005 \\ & 99999.0\mathrm{E}-00010 & \\ & 99999.0\mathrm{E}-00015 & \\ & 99999.0\mathrm{E}-00020 & \\ & 99999.0\mathrm{E}-00025 & \\ & 99999.0\mathrm{E}-00030 & \\ & 88000.0\mathrm{E}-00035 & \end{array}\right)$$

where $C_f^j$ and $C_m^j$ (resp.) are the *ex post* consumption matrix in floating-point arithmetics and multiple precision arithmetics (resp.).

29. - It was useless to display them in multiple precision arithmetics.

## *(ii) The parameterized Turbo-Pascal program*

```
(*******************************************************)
(*  PROGRAMME DE CALCUL D'UN TES PAR ITÉRATIONS JUSQU'À  *)
(*  LA CONVERGENCE ENTRE EMPLOIS ET RESSOURCES,          *)
(*  À UN CERTAIN SEUIL.                                  *)
(*******************************************************)

{$R-}    {Range checking off}
{$B+}    {Boolean complete evaluation on}
{$S+}    {Stack checking on}
{$I+}    {I/O checking on}
{$IFDEF CPU87}
{$N+}
{$ELSE}
{$N-}
{$ENDIF}
{$M 65500,16384,655360}

PROGRAM ITER_TES ;

USES DOS, CRT, UNIT_U ;

CONST Sizmax=5 ;

VAR ITERDAT :STRING ;
    TEST,Iter,Siz,Itermax :Integer ;
    DELTAP,DELTAP0,PROD,CHOC :Array[1..sizmax] of EXTENDED ;
    CT,CI :Array[1..sizmax,1..sizmax] of EXTENDED ;
    TEMP1,TEMP2,SEUIL :EXTENDED ;

BEGIN

{ INITIALISATION }
{ -------------- }
  for i :=1 to sizmax do begin
    PROD[i] :=0. ;
    CHOC[i] :=0. ;
    DELTAP[i] :=0 ;
    DELTAP0[i] :=0 ;
    for j :=1 to sizmax do begin
      CI[i,j] :=0. ;
      CT[i,j] :=0. ;
    end ;
  end ;
  ASSIGN(fx,'ITERTES.CFG') ;
  RESET(fx) ;
     readln(fx,ITERDAT) ;
     readln(fx,Itermax) ;
{%L} readln(fx,SEUIL) ;
  CLOSE(fx) ;
  ASSIGN(fx,ITERDAT) ;
  RESET(fx) ;
    readln(fx,siz) ;
    for i :=1 to siz do begin
      for j :=1 to siz do begin
{%L}    read(fx,CI[i,j]) ; { LECTURE DES CI }
      end ;
      readln(fx) ;
    end ;
```

```
    for i :=1 to siz do
      begin
{%L}    read(fx,PROD[i]) ; { LECTURE DE PROD }
      end ;
    readln(fx) ;
    for i :=1 to siz do
      begin
{%L}    read(fx,CHOC[i]) ; { LECTURE DE CHOC }
      end ;
    readln(fx) ;
  CLOSE(fx) ;
  for i :=1 to siz do
    begin
{%A}  DELTAP0[i] :=CHOC[i] ;
    end ;
  for i :=1 to siz do begin
    for j :=1 to siz do begin
{%C}  CT[i,j] :=CI[i,j]/PROD[j] ;
    { S_CT(i,j) ; }
    end ;
  end ;

  ASSIGN(fy,'ITERTES.OUT') ;
  REWRITE(fy) ;
    writeln(fy,'CALCUL DU TES PAR ITERATIONS') ;
    writeln(fy,'----------------------------') ;
    writeln(fy) ;
    writeln(fy,'TABLEAU DES CONSOMMATIONS INTERMEDIAIRES') ;
    writeln(fy,'----------------------------------------') ;
    for i :=1 to siz do begin
      for j :=1 to siz do fwrite(fy,CI[i,j]) ;
      writeln(fy) ;
    end ;
    writeln(fy) ;
    writeln(fy,'VECTEUR DE PRODUCTION') ;
    writeln(fy,'---------------------') ;
    for i :=1 to siz do fwrite(fy,PROD[i]) ;
    writeln(fy) ;
    writeln(fy) ;
    writeln(fy,'TABLEAU DES COEFFICIENTS TECHNIQUES') ;
    writeln(fy,'-----------------------------------') ;
    for i :=1 to siz do begin
      for j :=1 to siz do fwrite(fy,CT[i,j]) ;
      writeln(fy) ;
    end ;
    writeln(fy) ;
    writeln(fy,'VECTEUR DE CHOC') ;
    writeln(fy,'---------------') ;
    for i :=1 to siz do fwrite(fy,CHOC[i]) ;
    writeln(fy) ;
    writeln(fy) ;
    writeln(fy,'SEUIL DE CONVERGENCE') ;
    writeln(fy,'--------------------') ;
    fwrite(fy,SEUIL) ;
    writeln(fy) ;
    writeln(fy) ;
    writeln(fy) ;
    iter :=0 ;  { DÉBUT ITÉRATIONS }
    repeat
      iter :=iter+1 ;
      writeln(fy,iter :2,'E VAGUE') ;
```

```
      for i :=1 to siz do begin  { i=BRANCHE }
      { MISE A JOUR DES PRODUCTIONS }
      { -------------------------- }
{       PROD[i] :=PROD[i]+DELTAP0[i] ;      }
{       ----------------------------      }
{%C}    TEMP1 :=PROD[i]+DELTAP0[i] ;
{%A}    PROD[i] :=TEMP1 ;
        for j :=1 to siz do begin  { j=PRODUIT }
          gotoxy(30,10) ; write(j :3,'E PRODUIT') ;
{         DELTAP[i] :=DELTAP[i]+DELTAP0[j]*CT[i,j] ;     }
{         ----------------------------------------     }
{%C}      TEMP1 :=DELTAP0[j]*CT[i,j] ;
{%C}      TEMP2 :=DELTAP[i]+TEMP1 ;
{%A}      DELTAP[i] :=TEMP2 ;
        { MISE A JOUR DES CONSOMMATIONS INTERMEDIAIRES }
        { -------------------------------------------- }
{         CI[i,j] :=CI[i,j]+DELTAP0[j]*CT[i,j] ;         }
{         ------------------------------------         }
{%C}      TEMP1 :=DELTAP0[j]*CT[i,j] ;
{%C}      TEMP2 :=CI[i,j]+TEMP1 ;
{%A}      CI[i,j] :=TEMP2 ;
          fwrite(fy,DELTAP0[j]) ;
          write(fy,'*') ;
          fwrite(fy,CT[i,j]) ;
          if (j=siz) then
                      begin
                        write(fy,' = ') ;
                        fwrite(fy,DELTAP[i]) ;
                        writeln(fy) ;
                      end
                    else write(fy,'+') ;
        end ;
      end ;
      writeln(fy) ;
      TEST :=0 ;
      for j :=1 to siz do begin
        fwrite(OutPut,DELTAP[j]) ;
{%B}    if (ABS(DELTAP[j])<=SEUIL) then TEST :=TEST+1 ;
{%A}    DELTAP0[j] :=DELTAP[j] ;
{%V}    DELTAP[j] :=0 ;
      end ;
    until ((iter>=itermax) or (TEST=Siz)) ;
    writeln(fy,'NOUVEAU TABLEAU DES CONSOMMATIONS INTERMEDIAIRES') ;
    writeln(fy,'-----------------------------------------------') ;
    for i :=1 to siz do begin
      for j :=1 to siz do begin
        write(fy,CI[i,j],' ') ;
      end ;
      writeln(fy) ;
    end ;
    writeln(fy) ;
    writeln(fy,'NOUVEAU VECTEUR DE PRODUCTION') ;
    writeln(fy,'-----------------------------') ;
    for i :=1 to siz do begin
      write(fy,PROD[i],' ') ;
    end ;
    writeln(fy) ;
  CLOSE(fy) ;
END.
```

*(iii) - The floating-point arithmetic results listing*

```
CALCUL DU TES PAR ITERATIONS
----------------------------

TABLEAU DES CONSOMMATIONS INTERMEDIAIRES
----------------------------------------
150.000000 10.0000000 30.0000000
35.0000000 390.000000 80.0000000
15.0000000 100.000000 90.0000000

VECTEUR DE PRODUCTION
---------------------
300.000000 1000.00000 600.000000

TABLEAU DES COEFFICIENTS TECHNIQUES
-----------------------------------
0.50000000 0.01000000 0.05000000
0.11666667 0.39000000 0.13333333
0.05000000 0.10000000 0.15000000

VECTEUR DE CHOC
---------------
0.00000000 30.0000000 0.00000000

SEUIL DE CONVERGENCE
--------------------
0.00009000

1E VAGUE
0.00000000 * 0.50000000 + 30.0000000 * 0.01000000 + 0.00000000 * 0.05000000 = 0.30000000
0.00000000 * 0.11666667 + 30.0000000 * 0.39000000 + 0.00000000 * 0.13333333 = 11.7000000
0.00000000 * 0.05000000 + 30.0000000 * 0.10000000 + 0.00000000 * 0.15000000 = 3.00000000

2E VAGUE
0.30000000 * 0.50000000 + 11.7000000 * 0.01000000 + 3.00000000 * 0.05000000 = 0.41700000
0.30000000 * 0.11666667 + 11.7000000 * 0.39000000 + 3.00000000 * 0.13333333 = 4.99800000
0.30000000 * 0.05000000 + 11.7000000 * 0.10000000 + 3.00000000 * 0.15000000 = 1.63500000

3E VAGUE
0.41700000 * 0.50000000 + 4.99800000 * 0.01000000 + 1.63500000 * 0.05000000 = 0.34023000
0.41700000 * 0.11666667 + 4.99800000 * 0.39000000 + 1.63500000 * 0.13333333 = 2.21587000
0.41700000 * 0.05000000 + 4.99800000 * 0.10000000 + 1.63500000 * 0.15000000 = 0.76590000

NOUVEAU TABLEAU DES CONSOMMATIONS INTERMEDIAIRES
------------------------------------------------
1.50358500000000E+0002 1.04669800000000E+0001 3.02317500000000E+0001
3.50836500000000E+0001 4.08212220000000E+0002 8.06180000000000E+0001
1.50358500000000E+0001 1.04669800000000E+0002 9.06952500000000E+0001

NOUVEAU VECTEUR DE PRODUCTION
-----------------------------
3.00717000000000E+0002 1.04669800000000E+0003 6.04635000000000E+0002
```

### c - Appendix - Some `GNOMBR` algorithms and results

We present here the mathematical analysis of classical operator algorithms, we have made before implementing them in `GNOMBR`, then the comparison of factorial function computed between 20 and 25 to highlight the errors.

#### *(i) The classical operator's algorithms*[30]

Let's consider $X$, $Y$ operands, $\oplus, \ominus, \otimes, \oslash$ operators, $b$ the base and $\varepsilon_i$, $\delta_i$ carries over, we have $X = \sum_{i=q}^{m} x_i.b^i$ with $0 \leq x_i \leq b-1$ and $Y = \sum_{j=p}^{n} y_j.b^i$ with $0 \leq y_j \leq b-1$.

**The addition's algorithm**

$$X \oplus Y = \sum_{i=Inf(p,q)}^{Sup(n,m)+1} (x_i + y_i + \varepsilon_i - \delta_i).b^i$$

with $\varepsilon_i = 1$ if $(x_{i-1} + y_{i-1} + \varepsilon_{i-1} - \delta_{i-1}) > b-1$ and with $\varepsilon_i = 0$ else. On the other hand, we have $\delta_i = b$ if $(x_i + y_i + \varepsilon_i) > b-1$ and $\delta_i = 0$ else.

**The substraction's algorithm**

$$X \ominus Y = \sum_{i=Inf(p,q)}^{Sup(n,m)} (x_i - y_i - \varepsilon_i + \delta_i).b^i$$

with $\varepsilon_i = 1$ if $(x_{i-1} - y_{i-1}) < 0$ and $\varepsilon_i = 0$ else. On the other hand, we have $\delta_i = b$ if $x_i - y_i - \varepsilon_i < 0$ and $\delta_i = 0$ else.

**The multiplication's algorithm**

$$X \otimes Y = \sum_{r=0}^{n+m} (S_r + \varepsilon_r - \delta_r).b^r$$

where

$$S_r = \sum_{Inf(r,m)}^{k=Sup(0,r-m)} x_k * y_{r-k}$$

and $\varepsilon_i = 1$ if $S_{r-1} + \varepsilon_{r-1} + \delta_{r-1} > b-1$ else $\varepsilon_r = 0$ ; $\delta_r = b$ if $S_r + \varepsilon_r > b-1$ and $\delta_r = 0$ else.

**The division's algorithm**

Division is equivalent to $X = Y.Q + R$ where $Q$ is the quotient and $R$ the rest. We obtain the result of the division by iterations ; accuracy of division depends on it.

$$R_k = \sum_{i=0}^{p^k} (r_i^{k-1} - y_i).b^i$$

with $y_i = 0 \ \forall \ i / \ i \leq p^{k-1} - m$, the initial value is $R_0 = X$ and we have

$$Q_k = b^{p^{k-1}-m}$$

at the $k$-the iteration, we have two cases according to the value of $R_k$ :

⋄ if $R_k > Y$ then we do another iteration with

$$P^k = Sup_{i \in [0,n]}(i)$$

⋄ if $R_k \leq Y$ then we have

$$Q = \sum_{s=0}^{k} Q_s$$

30. - Implementation of arithmetical operators is in fact rather different in floating-point arithmetic from the classical one - see J.M.Muller (1989).

*(ii)* `GNOMBR` applied to computation of factorial function

**Table : Lost of accuracy with factorial function**

| | Multiple precision | Floating-Point | Difference |
|---|---|---|---|
| 20 ! | 2432.0E+00015 | 2432.0E+00015 | |
| | 90200.0E+00010 | 90200.0E+00010 | |
| | 81766.0E+00005 | 81766.0E+00005 | |
| | 40000.0E+00000 | 40000.0E+00000 | 0.0E+00000 |
| 21 ! | 51090.0E+00015 | 51090.0E+00015 | |
| | 94217.0E+00010 | 94217.0E+00010 | |
| | 17094.0E+00005 | 17094.0E+00005 | |
| | 40000.0E+00000 | | 40000.0E+00000 |
| 22 ! | 11.0E+00020 | 11.0E+00020 | |
| | 24000.0E+00015 | 24000.0E+00015 | |
| | 72777.0E+00010 | 72777.0E+00010 | |
| | 76076.0E+00005 | 76077.0E+00005 | |
| | 80000.0E+00000 | | -20000.0E+00000 |
| 23 ! | 258.0E+00020 | 258.0E+00020 | |
| | 52016.0E+00015 | 52016.0E+00015 | |
| | 73888.0E+00010 | 73888.0E+00010 | |
| | 49766.0E+00005 | 49766.0E+00005 | |
| | 40000.0E+00000 | | 40000.0E+00000 |
| 24 ! | 6204.0E+00020 | 6204.0E+00020 | |
| | 48401.0E+00015 | 48401.0E+00015 | |
| | 73323.0E+00010 | 73323.0E+00010 | |
| | 94393.0E+00005 | 94390.0E+00005 | 3.0E+00005 |
| | 60000.0E+00000 | | 60000.0E+00000 |
| 25 ! | 1.0E+00025 | 1.0E+00025 | |
| | 55112.0E+00020 | 55112.0E+00020 | |
| | 10043.0E+00015 | 10043.0E+00015 | |
| | 33098.0E+00010 | 33099.0E+00010 | |
| | 59840.0E+00005 | | -40160.0E+00005 |

## 3. - Conclusion

In our paper, we have recalled the accuracy problem, summary presented the main solutions available, and an application of our own multiple precision arithmetic tool to an input-output model. Such tools are promising although confidential. Indeed, the capabilities of computers are increasing so that, we can assert that we could still increase accuracy of scientific computations [31].

However, we have to specify that even if we could obtain a very high accuracy of the computation, it doesn't necessarily mean that we would have increased the number or significative digits in the results. If the significant digit's size of an observation is low, e.g. : 4 digits, even if we obtain a result with 20 digits, only 4 digits are significant [32]. We have only bounded the diffusion of floating-point round's error. Also, the main use of such arithmetics for macroeconomic or econometric models, is to decrease floating point error diffusion.

Nevertheless, we believe that two economic applications could use such tools to increase significant digits. Firstly, in the modelling of the Chaos Theory we could use such tools to determine the stability of systems according to the initial conditions [33]. Secondly, we believe that we could use multiple precision arithmetics to make counting process (e.g. : $A_n^p = \frac{n!}{(n-p)!}$ and $C_n^p = \frac{n!}{p!(n-p)!}$) to evaluate precisely the number of individuals in an Agent-Based computational or Artificial life large simulation.

31. - The only constraint is the size of the needed memory and the computation duration - see V.Ménissier, *Arithmétique exacte : conception, algorithmique et performances d'une implantation informatique en précision arbitraire*, Thèse de Doctorat, Université de Paris VI, 1994.

32. - About the general problem of error's measurement see J.Taylor (1996). About the economic observation and account, see O.Morgenstern (1950). As measurement, a datum has got an uncertainty part : the last digits. Jerrell M.E. (op.cit., 1997) has used arithmetic interval to decrease the effect of uncertainty on input-output models.

33. - About an overview of chaotic dynamics, see G.Abraham-Frois & E.Berrebi (*Instabilité, cycles, chaos,* Paris, Economica, 1995, pp.207-61). The authors explain that the logistic function used in the Day's Model (R.H.Day, "Irregular Growth Cycles", *American Economic Review*, 72, 1982, pp.406-14.), such as the following : $u_{n+1} = a.u_n.(1 - u_n)$, with $a = 3.6$ provides very different results because of the floating point round's error. Indeed, if we initiate the series with $u_1^a = 0.2$ and $u_1^b = 0.2000001$, after 10000 iterations the difference between $u_1^a$ and $u_1^b$ was more than 50 %. We have used `GNOMBR` to implement this logistic function and we observed that the mantissa of our program (1000 digits) was overflowed after 5 iterations.

These two ways could be taken, if we agree to increase accuracy memory size and duration computation[34]. But despite the advantage of technical progress, we still have to round some results, for example, when result is a number with an infinite digit's size (rational number, or from combination with $e$ or $\pi$).

## 4. - References

Alefeld G. & J.Herzberger, *Introduction to Interval Computations*, New York, Academic Press, 1983.

Avizienis A., "Signed Digit Number Representations for Fast Parallel Arithmetic", *IRE Transactions on Electronic Computers*, N°10, 1961, pp.389-400.

Berstel J., J.E.Pin & M.Pocchiola, *Mathématiques et informatique - tome 2 Combinatoire et arithmétique*, Paris, Ediscience international, Informatique, 1991, 257 p. + Programmes.

Buda R., "Présentation d'un outil de contrôle de la précision des calculs en modélisation macro-économétrique", *Document de travail GAMA*, Université de Paris X-Nanterre, août, 1996, 21 p. + Le logiciel `GNOMBR`.

———, "`SIMUL` - Manuel de références et guide d'utilisation version 3.1", *Document de travail GAMA*, Université de Paris X-Nanterre, 1999, 60 p. + Le logiciel `SIMUL`.

———, "Les algorithmes de la modélisation - une présentation critique pour la modélisation économique", *Document de travail Modem* N°01-44, Université de Paris X-Nanterre, juil., 2001, 97 p.

Daumas M. & J.M.Muller (Eds), *Qualité des calculs sur ordinateur - vers des arithmétiques plus fiables ?*, Paris, Masson, Informatique, 1997, 164 p.

Dumontet J., Vignes J., "Algorithme de dérivation numérique", *RAIRO*, Vol.1, 1989.

Jerrell M.E., "Interval Arithmetic for Input-Output Models with Inexact Data", *Computational Economics*, 1997, 10(1), pp.89-100.

Knuth D.E., *The Art of Programming - tome 2, Seminumerical Algorithms*, (Third ed.), Reading (Mass.), Addison-Wesley, 1997, 762 p.

Kulisch U., "Advanced Arithmetic for the Digital Computer, Design of Arithmetic Units", *Electronic Notes in Theoretical Computer Science*, 24 , Apr. 2000.a, 63 p.

———, "Interval Arithmetic in Forte Fortran", *Technical White Paper Sun Microsystems*, Palo Alto, 2000.b, 54 p.

34. - This size of our Turbo-Pascal software now depends on the RAM limit of the DOS, but it exist some techniques to remove this limit. Let's quote M.S.Khanniche & S.H.Yong ("A Solution to Memory Limit of DOS Based Large Finite Element Programs", *Advances in Engineering Software*, 21, 1994, pp.99-112). We have developed another algorithm (R.Buda, 2001). Anyway most of languages better manager memory than Turbo-Pascal.

Kulisch U., W.M.Miranker, *Computer Arithmetic in Theory and Practice*, New York, Academic Press, 1981.

La Porte M., Vignes J., Algorithmiques numériques - Tome 1, Paris, Techniq, 1974, 226 p.

Maillé M., "Some Methods to Estimate Accuracy of Measurements or Numerical Computations", *Processing of Mathematics for Computer*, Congress AFCET, 1982.

Morgenstern O., *On the Accuracy of Economic Observations*, Princeton, Princeton University Press, 1950.

Moore R.E., *Interval Analysis*, Prentice-Hall, Englewood Cliffs (N.J.), 1966.

Muller J.M., *Arithmétiques des ordinateurs - opérateurs et fonctions élémentaires*, Paris, Masson, Études et recherches en informatique, 1989, 214 p.

Pichat M. & J.Vignes, *Ingéniérie du contrôle de la précision des calculs sur ordinateurs*, Paris, Technip, Informatique, 1993, 233 p. + Programmes.

Sofronioua M. & G.Spaletta, "Precise numerical computation", *Journal of Logic and Algebraic Programming*, 64(1), Jul., 2005, pp.113-134.

Taylor J., *An Introduction to Error Analysis - : The Study of Uncertainties in Physical Measurements*, Enfield (New Hampshire), University Science Books, 1996, 327 p.

Wirth N., *Algorithmes et structures de données*, Paris, Eyrolles, 1987, 320 p.